Lisp and Scheme Presentation Notes

Grant Rettke grettke@acm.org

14 July 2009

1 Context

World Class Programmer, programmers, programs.

Immersion My approach for taking on non-trivial tasks

2 History

- Review the timeline
- Visit on the motivational differences: money versus education.
  - Note that Haskell turned out very differently; but perhaps only due to Miranda.

3 Similarities and Differences with Lisp

Similarities

- Just about everything
- People: Steele, Pitman, Gabriel, Nearly every Scheme implementer

Differences

- Separate namespaces for variables and functions (touch on lisp-2-ness re: "Technical Issues of Separation in Function Cells and Value Cell" and collisions in theory versus practice)
- Object system.
- `call/cc` (Note Graham implemented them in On Lisp
- Macros take syntax-objects as input and output rather than s-expressions.
- Tail calls in the spec

4 Styles

- Mostly functional (as noted by tail call requirement)
- OO
- Lazy
- Typed
- You get the picture

5 How to Learn Scheme

- TSPL
- R5RS

6 Culture

- Fierce independence
- Think for yourself
- Never trust a thought leader
- Attention to detail

7 Implementations

- Specifications: R3, R4, R5, R6
- Products: PLT, Chez, Gambit, Guile, Ikarus, MIT/GNU, Ypsilon, SISC, IronScheme.
- You tend to use more than just one

8 Libraries

- SRFIs

  1 List Library
  6 Basic String Ports
  13 String Library
  19 Time Data Types and Procedures
  26 Notation for Specializing Parameters without Currying
  28 Basic Format Strings
  39 Parameter objects
  42 Eager Comprehensions

- Individual downloads

9 IDEs

- VIM
- EMACS
- DrScheme
- SchemeWay (on Eclipse)
- xacc.ide
- LispIDE

10 Experiences

- A Crucible
- Deceptively simple
- Language power versus library power